8-19-05

$ AF
2194
IFW

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Guthrie et al.	Examiner:	Ho, Andy
Serial No.:	09/894,828	Group Art Unit:	2194
Filed:	06/29/2001	Docket:	40062/0250US01
Confirmation No.:	3720	Notice of Allow. Date:	n/a
Due Date:	08/22/2005		
Title:	ASP.NET HTTP RUNTIME		

CERTIFICATE UNDER 37 CFR 1.10:
"Express Mail" mailing label number: EV 118156916 US
Date of Deposit: August 17, 2005

I hereby certify that this paper or fee is being deposited with the U.S. Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to Mail Stop Appeal Brief-Patents, Commissioner for Patents P.O. Box 1450 Alexandria, Virginia 22313-1450.

By: Jenifer Weck
Name: Jenifer Weck

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

27488
PATENT TRADEMARK OFFICE

Sir:

We are transmitting herewith the attached:

☒ Transmittal Sheet in duplicate containing Certificate of Mailing
☒ Check(s) in the amount of $500 for filing fee for Appeal Brief
☒ Other: Appellant's Brief on Appeal in triplicate
☒ Return postcard

Please consider this a PETITION FOR EXTENSION OF TIME for a sufficient number of months to enter these papers or any future reply, if appropriate. Please charge any additional fees or credit overpayment to Deposit Account No. 13-2725. A duplicate of this sheet is enclosed.

Merchant & Gould P.C.
P.O. Box 2903
Minneapolis, MN 55402-0903
612.332.5300

By: [illegible]
Name: Tadd F. Wilson
Reg. No.: 54,544
TFWilson/jw

(PTO TRANSMITTAL - GENERAL)

OIPE JC42
AUG 1 7 2005
PATENT & TRADEMARK OFFICE

S/N 09/894,828 PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Guthrie, et al.	Examiner:	Ho, Andy
Serial No.:	09/894,828	Group Art Unit:	2194
Filed:	June 29, 2001	Docket No.:	40062.0250US01
Title:	ASP.NET HTTP RUNTIME		

CERTIFICATE UNDER 37 CFR 1.10:
"Express Mail" mailing label number: EV 118156916 US
Date of Deposit: August 17, 2005

I hereby certify that this paper or fee is being deposited with the U.S. Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to Mail Stop Appeals, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

By: [illegible]
Name: Jennifer Weck

APPELLANT'S BRIEF ON APPEAL

Mail Stop: Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This Brief is presented in furtherance of the Notice of Appeal filed June 22, 2005, from the final rejection of Claims 1-32 of the above-identified application, as set forth in the Final Office Action mailed January 12, 2005.

A check for $500.00 to cover the required fee under 37 CFR §41.20(b) for filing this Brief is enclosed.

Applicant reserves the right to request an oral hearing by filing a separate request for an oral hearing with the appropriate fee within two months of the date of the Examiner's Answer in response to this Brief.

08/22/2005 SHASSEN1 00000019 09894828
01 FC:1402 500.00 OP

This brief contains these items under the following headings, and in the order set forth below (37 C.F.R. §41.37(c)):

I. REAL PARTY IN INTEREST

The real party in interest in this appeal is Microsoft Corporation.

II. RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in this appeal:

☒ There are no such appeals or interferences.

☐ These are as follows:

III. STATUS OF CLAIMS

The status of the claims in this application are:

A. TOTAL NUMBER OF CLAIMS IN APPLICATION: 32

B. STATUS OF ALL THE CLAIMS:

1. Claims cancelled: None
2. Claims withdrawn from consideration but not cancelled: None
3. Claims pending: 1-32
4. Claims allowed: None
5. Claims rejected: 1-32

C. CLAIMS ON APPEAL: Claims 1-32

IV. STATUS OF AMENDMENTS

An amendment after final was filed subsequent to the final rejection, but the Examiner did not enter the amendment for the purposes of appeal.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The present invention is generally directed to methods for caching data for use in dynamically generated web pages. See, e.g., Figures 1 and 4, and page 15, line 15 to page 16, line 2. A concise explanation of each independent claim is provided below:

A. Independent Claim 1

Independent claim 1 generally relates to a computer runtime for handling a Hypertext Transfer Protocol (HTTP) request. See, e.g., Figure 3 and paragraphs [0023] and [0024]. The runtime includes a context object that logically represents a received HTTP request and that encapsulates at least one property of the received request. See, e.g., Figure 3 and paragraphs [0024] and [0026]. The runtime also includes an event pipeline corresponding to the context object for processing the context object. See, e.g., Figure 3 and paragraphs [0024] and [0025]. The event pipeline includes a plurality of request events such that when an event (corresponding to a request event) occurs a call-back is generated. See, e.g., Figure 3 and paragraphs [0024] and [0025]. The call-back initiates each application and each module that is registered for the request event to process the context object. See, e.g., Figure 3 and paragraphs [0026] and [0027].

B. Independent Claim 10

Independent Claim 11 generally relates to a method for processing an HTTP request. See, e.g., Figure 3 and paragraphs [0023] and [0024]. The method includes forming a context object that logically represents a received HTTP request and that encapsulates at least one property of the received request. See, e.g., Figure 3 and paragraphs [0024] and [0026]. The method further includes forming an event pipeline corresponding to the context object for processing the context object. See, e.g., Figure 3 and paragraphs [0024] and [0025]. The event pipeline includes a plurality of request events, each request event corresponding to an event. See, e.g., Figure 3 and paragraphs [0024] and [0025]. The method includes generating a call-back when the event corresponding to a request event is raised and when at least one of an application and a module is registered in association with the request event. See, e.g., Figure 3 and paragraphs [0024] through [0027]. The method further includes initiating each application and each module that is registered in association with the request event in response to the callback for processing the context object. See, e.g., Figure 3 and paragraphs [0026] through [0027].

C. Independent Claim 22

Independent claim 22 is independent claim 11 rewritten into Beauregard form. Independent claim 22 generally relates to a computer-readable medium having computer-executable instructions for processing a Hypertext Transfer Protocol (HTTP) request. See, e.g., Figure 1 and paragraph [0018]. The processing method includes forming a context object that logically represents a received HTTP request and that encapsulates at least one property of the received request. See, e.g., Figure 3 and paragraphs [0024] and [0026]. The processing method further includes forming an event pipeline corresponding to the context object for processing the context object. See, e.g., Figure 3 and paragraphs [0024] and [0025]. The event pipeline includes a plurality of request events, each request event corresponding to an event. See, e.g., Figure 3 and paragraphs [0024] and [0025]. The processing method includes generating a call-back when the event corresponding to a request event is raised and when at least one of an application and a module is registered in association with the request event. See, e.g., Figure 3 and paragraphs [0024] through [0027]. The processing method further includes initiating each application and each module that is registered in association with the request event in response to the callback for processing the context object. See, e.g., Figure 3 and paragraphs [0026] through [0027].

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The Examiner rejected claims 1-32 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Gosling (USPN 6,247,044) in view of Logston (USPN 6,687,735). In order to establish *prima facie* obviousness under 35 U.S.C. 103(a), three basic criteria must be met, namely: (1) there must be some suggestion or motivation to combine the references or modify the reference teaching; (2) there must be a reasonable expectation of success; and (3) the reference or references when combined must teach or suggest each claim limitation (Manual of Patent Examining Procedure 2142). Applicants submit that the Office Action failed to state a *prima facie* case of obviousness, and therefore the burden has not properly shifted to Applicants to present evidence of nonobviousness. Applicants respectfully assert that the Examiner has failed to establish a *prima facie* case of obviousness because the references fail to disclose or suggest all of the limitations of the pending claims as discussed herein.

Specifically, the grounds of rejection presented for review are:

1. With respect to all claims, whether Gosling teaches or discloses the claimed element "forming an event pipeline corresponding to the context object."

2. With respect to all claims, whether Gosling or Logston, alone or in combination, teach or disclose the claimed element of an "event pipeline having a plurality of request events" in which "each request event having a corresponding event."

3. With respect to all claims, whether Gosling teaches away from using an "event pipeline" as claimed.

4. With respect to all claims, whether Logston teaches or discloses "generating a call-back event."

5. With respect to all claims, whether Logston teaches or discloses "initiating each application and each module that is registered in association with the request event."

VII. ARGUMENT

A. NEITHER GOSLING NOR LOGSTON ANTICIPATE "FORMING AN EVENT PIPELINE"

The Examiner cites Gosling, Col. 5, lines 42-43 as anticipating the "forming an event pipeline corresponding to the context object" element of the claimed invention. Final Office Action (mailed January 12, 2005), Page 3, lines 3-8. Applicants respectfully disagree with the Examiner's analysis of Gosling.

Gosling is directed to a system in which a plurality of servlet objects is used to replace the use of a monolithic server application. Gosling specifically states that the servlets are continuously operating and waiting for calls from the server computer. See, Gosling, Col. 2, lines 8-11 ("The servlet objects operate in a continual loop until invoked. Thus, there is no startup overhead associated with execution of the servlet objects."). See also, Gosling, Col. 3, lines 17-30 ("Each servlet 56 is an instantiated software object waiting to be invoked... However, unlike a CGI script, a servlet object of the present invention is instantiated at server start-up. Thus, the servlet can be thought of as operating in a continual loop waiting to be executed. Observe that after instantiation there is no computational start-up expense when the servlet is called."). The Examiner's citation only applies to the situation in which a necessary servlet is not locally available. See, Gosling Col. 5, lines 40-55. In addition, Gosling goes on to state that, once instantiated, the servlets loop continuously until the server is shut off or a destroy method is called. See, Col. 6, lines 45-46 ("Once instantiated, a servlet loops until the server is shut off or a destroy method is called on the servlet by the server.")

As the above discussion shows, in Gosling none of the servlets correspond to a context object, or even an HTTP request. The servlets (even those retrieved from a remote server as necessary), once instantiated, wait to be called regardless of the requests that are received. While servlets may correspond to a request type, they do not correspond to any specific request or context object. Thus, a servlet cannot be considered to correspond directly to a specific request, and therefore do not anticipate an event pipeline corresponding to a context object.

It should be noted that Logston similarly does not disclose an "event pipeline corresponding to a context object." Logston discloses a distributed application in which a client portion (Logston's DACP) and a server portion (Logston's DASP) work in concert as a single application, the benefit of which is to minimize the amount of code transmitted to the client, the work done at the client, and allow easy load-balancing at the server by allowing multiple identical DASPs to be allocated throughout a server farm as necessary. Similar to Gosling, Logston's DASPs exist independently of the DACPs and, indeed, specific DACP requests with the exception that a DASP will be shutdown if it is not servicing (i.e. registered for) at least one DACP.

For the reasons given above, Applicants believe that neither Gosling nor Logston, singly or in combination anticipate the event pipeline element of the independent claims in the pending application. Therefore, Applicants respectfully request that the Examiner withdraw his rejection and find claims 1-32 in a condition for allowance.

B. NEITHER GOSLING NOR LOGSTON ANTICIPATE AN "EVENT PIPELINE HAVING A PLURALITY OF REQUEST EVENTS" OR "REQUEST EVENT HAVING A CORRESPONDING TO EVENT"

The Examiner cites Gosling, Col. 7, line 15 as anticipating the "event pipeline having a plurality of request events" element of the claimed invention. Final Office Action, Page 3, lines 3-8. Applicants respectfully disagree with the Examiner's analysis of Gosling and the invention as claimed.

It appears from the citation provided that the Examiner is equating an HTTP request in Gosling with a request event in an event pipeline in the claimed invention. The present application states in paragraph [0025] that, "The event pipeline provides an event-based architecture that includes a plurality of request events for generating a callback when the request event is raised." Thus, the request events are part of the event pipeline formed corresponding to the context object and not necessarily part of the actual request received.

The Applicants believe that the Examiner is trying to interpret Gosling's servlet as an event pipeline (which is incorrect for the reasons given above) and then cannot provide support for the distinguishing elements of the event pipeline. This is probably because Gosling states that each HTTP request is mapped to a single servlet. See, Gosling, Col. 4, lines 46-49 ("The thread then maps the request to a servlet name (step 78). The servlet may be specified by a URL, in which case the mapping process is direct."). While this servlet may initiate calls to other servlets, Gosling is clear that each HTTP request is mapped to a single servlet. The Applicants' system, on the other hand, forms the event pipeline that then uses as many applications and modules as necessary to process the request events in the event pipeline raised when processing the context object.

For at least these reasons, Applicants believe that Gosling does not anticipate an "event pipeline having a plurality of request events" and each "request event having a corresponding event" as claimed. Therefore, Applicants respectfully request that the Examiner withdraw the rejection of the independent claims 1, 11 and 22 and all the claims that depend therefrom.

C. GOSLING TEACHES AWAY FROM "FORMING AN EVENT PIPELINE"

This may already be clear from the above discussion, but Gosling clearly teaches away from forming an event pipeline as claimed and as argued by the Applicants in their various responses. In numerous locations, Gosling states that the benefit of his system is that the servlets are already running upon receipt of client requests. See, e.g., Gosling, Col. 2, lines 10-11 ("Thus, there is no startup overhead associated with execution of the servlet objects."). See also, Gosling, Col. 3, lines 27-30 ("a servlet object of the present invention is instantiated at server start-up. Thus, the servlet can be thought of as operating in a continual loop waiting to be executed. Observe that after instantiation there is no computational start-up expense when the servlet is called.").

Gosling would consider forming an event pipeline for each context object formed to be an unreasonable computing overhead because of the additional computation resources necessary for each request. Gosling would categorically object to the Applicants' system as disclosed in the specification in paragraph [0025] in which "Each HTTP Module that is registered with one or more request events is instantiated when the pipeline is instantiated." Gosling's premise is to have everything instantiated prior to the receipt of the HTTP request. In at least this aspect Gosling's system is diametrically opposed the disclosed system of the Applicants.

D. LOGSTON FAILS TO ANTICIPATE "GENERATING" AND "INITIATING" AS CLAIMED

The Examiner asserts that, while Gosling does not disclose or suggest the "generating" or "initiating" elements as claimed, these elements may be found in Logston. The Examiner presents Logston as anticipating the "generating" and "initiating" elements of the claimed invention. The Applicants respectfully disagree.

Logston discloses a distributed application in which a client portion (Logston's DACP) and a server portion (Logston's DASP) form a single application, the benefit of which is to minimize the amount of code transmitted to the client, to minimize the amount of work done at the client, and to allow easy load-balancing at the server by allowing multiple identical DASPs to be allocated throughout a server farm as necessary. The important structural elements of Logston include that:

- The DACP and DASP are elements of a single application (Col. 7, lines 57-67);
- The DACP is provided by the server to the client as part of the startup process (Col. 8, lines 17-19);
- Every DASP is identical (Col. 8, lines 1-16);
- Each DACP must be assigned (registered) to a DASP for **all** communications from the DACP (Col. 33, lines 21-27);
- Each DASP is also "registered" with the OS to receive requests from the OS (Col. 33, 10-13) as is well known for any application that must interact with the OS;
- A DASP can service multiple DACPs (Col. 8, lines 1-16);
- A DASP remains in existence from the initial assignment to a DACP until either there is a timeout period during which there are no communications received

from a DACP or the last DACP terminates (Col. 34, lines 3-51). Thus, Logston's DASPs may operate forever, under a scenario in which new DACPs are assigned to the DASP as old DACPs terminate.

Applicants believe that the citations provided by the Examiner relate only to the proposition that communication from a client (i.e., the DACP) can be directed to the appropriate DASP running on the server and that the DASP can receive requests from the operating system or other global control programs. This is well known in the art and is a fundamental part of the client server model. That Logston uses the same terminology in "registered" and "callback" does not mean that Logston has anticipated the claimed elements.

Applicants believe that Logston does not disclose the "generating" and "initiating" elements of the claimed invention for several reasons. First, the "generating" claim element includes the requirement that the "at least one of an application and a module is registered in association with <u>the request event</u>." Logston's DASP (while capable of receiving communications from any DACP) is registered for specific DACPs to receive all communications from those DACPs. Logston's DASPs are not registered in association with a specific request event that may be raised by specified request.

Furthermore, Logston does not anticipate the "initiating" element in the claimed invention as that element includes the claimed limitation that each application/module initiated be "registered in association with <u>the request event</u>..." Even at Logston's initial startup, Logston does not initiate a module registered in association with a request event of an event pipeline. Rather, Logston initiates a DASP registered in association with <u>a specified client portion DACP</u>.

The Examiner may be arguing that the DASP's registration with the operating system anticipates the claimed language, but this is also in error because the argument does not take into account that the request events of the claimed invention are part of the event pipeline that correspond to events that may be raised by a HTTP request. The events disclosed in Logston relate only to the operating system or the OpenCable application, neither of which are events that may be raised by a HTTP request when processed by an event pipeline formed for a context object.

In sum, while Logston does disclose registering DASPs for DACPs and registering a DASP with its operating system, Logston does not disclose registering anything in association with a request event, much less a request event associated with an event pipeline formed corresponding to a context object. For at least these reasons, Logston does not teach or disclose the claimed elements of "generating" or "initializing" with all their limitations. Therefore, the Applicants respectfully request that the rejection be withdrawn and all pending claims be allowed.

VIII. CONCLUSION

As described above, Applicants have shown that the Examiner has failed to establish a *prima facie* case of obviousness. Therefore, it is earnestly requested that the Examiner's rejections be reversed, and that all of the pending claims be allowed.

Respectfully submitted,

Dated: August 17, 2005

[illegible]

SIGNATURE OF ATTORNEY

27488
PATENT TRADEMARK OFFICE

Tadd F. Wilson, Attorney Reg. No. 54,544
MERCHANT & GOULD P.C.
P. O. Box 2903
Minneapolis, MN 55402-0903
(303) 357-165`

IX. CLAIMS APPENDIX

The text of the claims involved in the appeal are:

Listing of Claims:

1. (original) A Hypertext Transfer Protocol (HTTP) request handling runtime, comprising:

a context object logically representing an HTTP request that is received at a host application from a client application, the context object encapsulating at least one property associated with the received HTTP request; and

an event pipeline corresponding to the context object, the event pipeline having a plurality of request events, each request event having a corresponding event and generating a call-back when the event corresponding to the request event is raised and when at least one of an application and a module is registered in association with the request event, each call-back initiating each application and each module that is registered in association with the request event to process the context object.

2. (original) The HTTP request handling runtime according to claim 1, wherein the plurality of request events have a deterministic order.

3. (original) The HTTP request handling runtime according to claim 2, wherein at least one of the plurality of request events is a synchronous request event.

4. (original) The HTTP request handling runtime according to claim 2, wherein at least one of the plurality of request events is an asynchronous request event.

5. (original) The HTTP request handling runtime according to claim 2, wherein the plurality of request events further includes at least one request event having a non-deterministic order.

6. (original) The HTTP request handling runtime according to claim 1, wherein the plurality of request events have a non-deterministic order.

7. (original) The HTTP request handling runtime according to claim 6, wherein the plurality of non-deterministic order request events include an error event.

8. (original) The HTTP request handling runtime according to claim 1, wherein a module is registered in association with a plurality of request events.

9. (original) The HTTP request handling runtime according to claim 1, wherein the event pipeline is a separate instance of the event pipeline for each HTTP request that is received at the host application from a client application.

10. (original) The HTTP request handling runtime according to claim 1, wherein HTTP request runtime parses the received HTTP request to form the context object that logically represents the HTTP request.

11. (original) A method for processing a Hypertext Transfer Protocol (HTTP) request, comprising the steps of:

forming a context object that logically represents an HTTP request that is received at a host application from a client application, the context object encapsulating at least one property associated with the received request;

forming an event pipeline corresponding to the context object, the event pipeline having a plurality of request events, and each request event having a corresponding event;

generating a call-back when the event corresponding to a request event is raised and when at least one of an application and a module is registered in association with the request event; and

initiating each application and each module that is registered in association with the request event in response to the callback for processing the context object.

12. (original) The method according to claim 11, further comprising a step of registering a module in association with at least one selected request event.

13. (original) The method according to claim 11, further comprising a step of registering a plurality of modules in association with a selected request event.

14. (original) The method according to claim 11, wherein the plurality of request events have a deterministic order.

15. (original) The method according to claim 14, wherein at least one of the plurality of request events is a synchronous request event.

16. (original) The method according to claim 14, wherein at least one of the plurality of request events is an asynchronous request event.

17. (original) The method according to claim 16, wherein the plurality of request events further includes at least one request event having a non-deterministic order.

18. (original) The method according to claim 11, wherein the plurality of request events have a non-deterministic order.

19. (original) The method according to claim 18, wherein the plurality of non-deterministic order request events include an error event.

20. (original) The method according to claim 11, wherein the step of forming the event pipeline corresponding to the context object forms the event pipeline as a separate instance for each HTTP request received at the host application from a client application.

21. (original) The method according to claim 11, wherein the step of forming the context object includes a step of parsing the received HTTP request to form the context object.

22. (original) A computer-readable medium having computer-executable instructions for processing a Hypertext Transfer Protocol (HTTP) request comprising steps of:

forming a context object that logically represents an HTTP request that is received at a host application from a client application, the context object encapsulating at least one property associated with the received request;

forming an event pipeline corresponding to the context object, the event pipeline having a plurality of request events, and each request event having a corresponding event;

generating a call-back event when the event corresponding to a request event is raised and when at least one of an application and a module is registered in association with the request event; and

initiating each application and each module that is registered in association with the request event in response to the callback for processing the context object.

23. (original) The computer-readable medium according to claim 22, further comprising a step of registering a module in association with at least one selected request event.

24. (original) The computer-readable medium according to claim 22, further comprising a step of registering a plurality of modules in association with a selected request event.

25. (original) The computer-readable medium according to claim 22, wherein the plurality of request events have a deterministic order.

26. (original) The computer-readable medium according to claim 22, wherein at least one of the plurality of request events is a synchronous request event.

27. (original) The computer-readable medium according to claim 22, wherein at least one of the plurality of request events is an asynchronous request event.

28. (original) The computer-readable medium according to claim 25, wherein the plurality of request events further includes at least one request event having a non-deterministic order.

29. (original) The computer-readable medium according to claim 22, wherein the plurality of request events have a non-deterministic order.

30. (original) The computer-readable medium according to claim 29, wherein the plurality of non-deterministic order request events include an error event.

31. (original) The computer-readable medium according to claim 22, wherein the step of forming the event pipeline corresponding to the context object forms the event pipeline as a separate instance for each HTTP request received at the host application from a client application.

32. (original) The computer-readable medium according to claim 22, wherein the step of forming the context object includes a step of parsing the received HTTP request to form the context object.

X. EVIDENCE APPENDIX

None

XI. RELATED PROCEEDINGS APPENDIX

None